Hierachical Generation of Pin accurate SystemC Models based on RF Circuit Schematics

Yifan Wang
Chair of Integrated Analog Circuits and RF Systems
Sommerfeldstrasse 24
RWTH Aachen, Germany
wang@ias.rwth-aachen.de

Zhimiao Chen
Chair of Integrated Analog Circuits and RF Systems
Sommerfeldstrasse 24
RWTH Aachen, Germany
ias@rwth-aachen.de

Stefan Heinen
Chair of Integrated Analog Circuits and RF Systems
Sommerfeldstrasse 24
RWTH Aachen, Germany
ias@rwth-aachen.de

ABSTRACT
This paper presents a method to generate pin accurate SystemC models based on the RF circuit schematics, using a SKILL [1] based routine. This method addresses the verification and modeling of the analog/RF circuits in the event driven digital domain using SystemC, based on the same data base created by the circuit designer. The proposed method was employed to model a specific test case, comprising a phase locked loop in a RF receiver chain. The reliability of the SystemC models have been proven by comparing the simulation results with the Verilog-AMS(wreal) models. The pin accurate SystemC models were found to be well suited for top down design, virtual prototyping and verification in SoC implementation. Additionally, the SystemC models are very efficient regarding the simulation speed and flexibility to pass abstract data types with real numbers through the ports.

Keywords
SystemC, RF, modeling, verification, virtual prototyping SoC

1. INTRODUCTION
Verification is an indispensable task from the virtual prototyping to the verification phase in today's design of complex, highly integrated circuits and System on Chips. Considering the functional verification case, it is almost impossible to verify the whole system at the transistor level, since one simulation run would take days or even weeks to complete. Thus modeling becomes the most critical and crucial part of the verification task. Considering the system level design phase, most models have been created by using Matlab or C-based HDLs, like SystemC [2]. SystemC is a hardware description language built on top of C++, which has the capability to reuse existing C++ codes or libraries. Furthermore, SystemC exhibit all the object-oriented features of C++, offers more flexibility and portability of the models. Therefore the consistency between the System level model and the circuit level data base will also be required.

A lot of works have been done to develop an efficient way to model and simulate the analog/RF sub systems in order to coming up with the increasing complexity and verification coverage requirement. The results lead to map the analog specification either by using equivalent base band models of the RF circuits [3] [4], or by passing real number traffic between the models in the digital simulator [5] [6], or by combining the two aforementioned approaches with a custom Verilog PLI function in order to get the most simulation performance [7]. Most of the models of the analog/RF subsystems are hand crafted, therefore the modeling and verification team has to spend a lot of time to update and validate the models to ensure the design changes have been also covered by the models. If the model doesn't capture the essential behavioral of the analog circuits due to outdated data base or inaccurate pin definition, the verification will fail.

The proposed method intents to refine system level models based on the schematic information, this leads to the possibility to keep the consistency between the system level models and those generated from schematic level. On the one hand, the connectivity information of the analog/RF sub system can be extended into the system level and verified, therefore the consistent virtual RF prototype can be provided. On the other hand, the behavioral models in SystemC can be executed more efficiently, thus helps the development of further system components as well as SoC verification.

The remainder of this paper describes the basic idea of the generation of pin accurate SystemC models based on the Schematic data base. Section 3 demonstrates the results of this approach applied on a PLL system and discusses the difference about behavioral model generation in SystemC and SystemC-AMS. The simulation results of the SystemC models has been compared to the results of the models implemented using Verilog-AMS wreal approach, which has been shown in [6]. Furthermore, the extension of the PLL output signal to the base band signal and it interaction with both the signals in the RF chain and detailed pass band PLL model regarding the system level simulation, verification and virtual prototyping will be discussed.

2. MODEL GENERATION AND IMPLEMENTATION

Due to the computational capability constrains, the virtual prototyping and functional verification of the RF subsystems have to be done in a digital simulator, in order to get the most simulation efficiency and to achieve the co-Simulation feasibility in the whole SoC. Thus the automatic generation of pin accurate SystemC RF models has been implemented. Two possibilities have been considered in order to get the structure information of the existing circuits/subsystems and export it accordingly. 1.) A pure netlist translator, which parses the whole netlist into SystemC bodies, this kind of translator need two steps to generate pin accurate models: first, it has to build up the hierarchy from the flat netlist, which has been generated from the schematics. Second, generate the SystemC body of the models and compare the schematic instance with existing instances in the SystemC data base. While this approach is generic and almost universal for every SPICE netlist type, it needs intensive netlist keyword extension, since a mixed-signal netlist always contains analog/digital netlists, simulation controls and HDL behavioral models. Additionally, the data flow direction in the SystemC models has to be defined based on the circuit information, since the models in SystemC don't have any graphical representation like the schematic in order to reduce the error-proneness during the modeling process. 2.) A direct translator of the given schematic using the SKILL [1] language. Since the schematic already comprises the hierarchy structure of the given System, the SKILL language features the possibility to translate this directly into SystemC. Additionally, the property of the ports can be invoked from SKILL to determine the I/O direction of the ports in the equivalent, pin accurate SystemC models. Figure 1 shows the simplified flow chart of the proposed SystemC model generation routine.

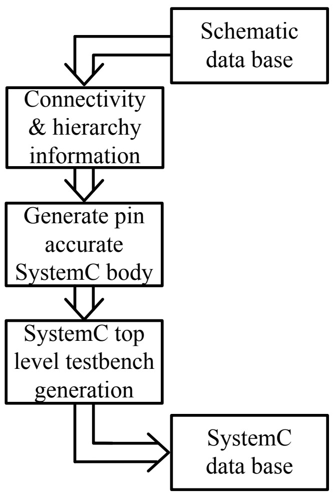
Schematic data base
Connectivity & hierarchy information
Generate pin accurate SystemC body
SystemC top level testbench generation
SystemC data base

Figure 1: Simplified flow chart of the SystemC model generation process.

The SKILL routine contains two procedures: the first part collects the structure information of the circuits, inclusive their hierarchical levels, instance names, pins and port names. This information is used to generate the equivalent SystemC models. The basic structure of the models will be created with the port definition and hierarchical connectivity. A part of the SKILL code is shown in the listing 1:

```
procedure(SCModuleOutput(subport gotView header)

fprintf(subport sprintf(nil ”\nSC_MODULE(%s){\n\n”
header~>cellName))
;search all pins
foreach(pin gotView~>terminals
;define SC_port based on pin~>direction
if(pin~>direction == ”input” then
fprintf(subport ”sc_in<bool>%s\n” pin~>name)
)
if(pin~>direction == ”output” then
fprintf(subport ”sc_out<bool>%s\n” pin~>name)
)
if(pin~>direction == ”inout” then
fprintf(subport ”sc_inout<bool>%s\n” pin~>name)
))

;search all connections
foreach(NetConnc gotView~>nets~>name

;define all sc_signal used to
;connect pins from different components
if(!member(NetConnc switchedView~>terminals~>name)
fprintf(subport sprintf(nil ”sc_signal<bool>
%s\n” NetConnc))))

fprintf(subport sprintf(nil ”void␣sig_proc(){”))
fprintf(subport ”␣};”)
fprintf(subport sprintf(nil
”SC_CTOR(%s){\n␣SC_METHOD(sig_proc);\n\n”
header~>cellName))

;search all instances(components)
;write signal definition
)
```

Listing 1: Part of the SKILL code to export the SC_module bodies based on the schematic information.

After the equivalent models of the instances have been created, the second procedure generates the test bench based on the top level test bench information from the schematic level. Since some of the instances like ”sheets” are not needed, a list has been implemented in order to exclude the instances, which is not necessary for the modeling process.

```
let(
cv = geGetEditCellview()
;define Exclude_List, SC modules of the
;instances in the List wont be generated
Exclude_List = list(...)
;call for Schecmaticstruct()
;print sc_signal in testcase.cpp
foreach(NetConnc cv~>nets~>name
fprintf(mainport sprintf(nil
”sc_signal<bool>␣%s\n” NetConnc)))

;describe connections in testcase.cpp
if(!member(header~>libName Exclude_List) then
index = 0
foreach(inst header~>instances
if(!member(inst~>cellName
list(”ipin” ”opin” ”iopin”)) then
fprintf(mainport sprintf(nil ”%s␣i%d_%s(\”%s\”);\n”
inst~>cellName index
inst~>cellName inst~>cellName))
foreach(Terminal inst~>terminals
fprintf(mainport sprintf(nil ”i%d_%s.%s(%s);\n”
index inst~>cellName
Terminal~>name Terminal~>net~>name)))
fprintf(mainport ”\n”)
index = index + 1))

;print simulation control in testcase.cpp
;view testcase.cpp files
t)
```

Listing 2: Part of the SKILL code to generate the SystemC top level test bench based on the schematic information.

The complete SystemC model set can be created by executing both of the SKILL routines recursively. An additional parameter has been created in order to define the abstraction level of the model generation. Since the models are mostly used for the system/block level model refinement and virtual prototyping, transistor level information is not in the main focus of this approach. The model generator only creates the body of the SystemC model with connection information, the behavioral of the models has to be implemented manually. The simulation of the SystemC models can only be done in the time domain, thus some of the analog specifications in the frequency domain like phase noise and filter characteristics have to be mapped into the event driven time domain, which has been discussed in [6].

3. TEST CASE: PHASE LOCKED LOOP

The test case presented in this paper consists a fractional N phase locked loop (PLL) system, which has been implemented, verified and brought to tape out. The proposed SystemC model generation approach has been applied on the PLL system shown in figure 2. Based on circuit level simulation, the PLL has following specifications:

VDD	$1.0V$
f_{out}	$900\ MHz - 1\ GHz$
f_{ref}	$200MHz$ or $25MHz$
$\Delta\Sigma$ modulator	$3.rd$ Order
Phase noise	$-100\ dBc/Hz$ @ $\Delta f =\ 1MHz$

Table 1: The specification of the PLL based on circuit level simulation.

The model refinement has been initiated after the hierarchical extraction of the pin accurate PLL models in SystemC.

3.1 The SystemC PLL Model

The behavioral models in SystemC have been implemented based on the event driven approach, since SystemC only supports the description of hardware by using discrete event model of computation [2]. The phase noise mapping in the VCO has been realized by the method talked in [6]. In addition, the generation of the normal distributed jitter noise has been implemented base on the box-muller algorithm. part of the VCO code has been shown in listing 3.

```
//***automatic generated part:
SC_MODULE(block_3stage_vco_calibration_dnw){
   sc_in<double>VCON
   sc_in<bool>HIGHER_FREQ
   sc_in<bool>FAST_CALIBRATION
   sc_in<bool>SLOW_CALIBRATION
   sc_in<double>VSS
   sc_in<double>VDD
   sc_out<BB_SIG>VCO_OUT
//other hierarchical connection inforamtion
...
//***hand crafted part
//define the model parameters
double v_min, v_max, kvco, freq_min;
double phasenoise, deltaf, std, mean, jitter;

void sig_proc(){
//calculate the frequency
freq = (pin_VCON.read()-v_min)*kvco +freq_min;
//calculate the phase noise
//box muller to generate normal distribution
bool used_last=0;
double x1, x2, y1, y2, w;
if(used_last==1){
y1=y2;
used_last=0;
}else{
do{
x1=2.0*(((double)rand())/((double)RAND_MAX)) - 1.0;
x2=2.0*(((double)rand())/((double)RAND_MAX)) - 1.0;
w=x1*x1+x2*x2;
}while(w>=1.0);
w=sqrt((-2.0*log(w))/w);
y1=x1*w; y2=x2*w;
used_last=1;
}

//calculate the equivalent timing jitter
next = now + 0.5/freq + Jitter;
//write out the VCO signal
VCO_OUT.write(...);
};
```

Listing 3: Part of the SystemC VCO source code.

Listing 4 shows he automatic generated top level test bench of the PLL based on circuit level information.

```
int sc_main(int argc, char* argv[]){
sc_signal<bool> net028, net027;
sc_signal<bool> vco_tune[2];
sc_signal<double> vdd;
sc_signal<BB_SIG> VCO_outn;
sc_signal<bool> net48;
sc_signal<bool> net048
sc_signal<bool> freq_out
sc_signal<double> gnd
sc_signal<bool> net61, net043, net9;
sc_signal<bool> net023, net037, net050;
sc_signal<BB_SIG> VCO_outp;
sc_signal<double> net047;
sc_signal<bool> net026;

PLL_toplevel i0_PLL_toplevel("PLL_toplevel");
i0_PLL_toplevel.VCO_Out_N(VCOoutn);
i0_PLL_toplevel.VCO_Out_P(VCOoutp);
i0_PLL_toplevel.F_Reference(net61);
i0_PLL_toplevel.Fast_calibration(vco_tune[1]);
i0_PLL_toplevel.Ibias_100u_source(net047);
i0_PLL_toplevel.Slow_calibration(vco_tune[0]);
i0_PLL_toplevel.Switch_Reference(net028);
i0_PLL_toplevel.VDD(vdd);
i0_PLL_toplevel.VSS(gnd);
i0_PLL_toplevel.frac_mode(net027);
i0_PLL_toplevel.reset(net026);
i0_PLL_toplevel.sdin(net9);

digital_config_interface
i0_("digital_config_interface");
i0_.DIV_RATIO(net9);
i0_.DS_RST(net026);
i0_.FRAC_MODE(net027);
i0_.REF_SEL(net028);
i0_.VCO_TUNE(vco_tune);
i0_.CFG_DATA(net037);
```

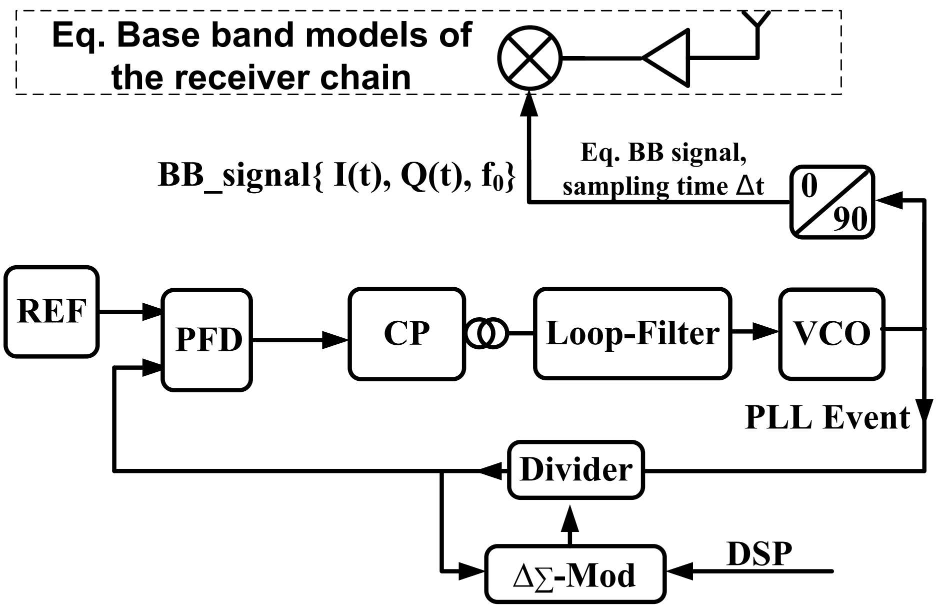
Eq. Base band models of the receiver chain
BB_signal{ I(t), Q(t), f0}
Eq. BB signal, sampling time Δt
0
90
REF
PFD
CP
Loop-Filter
VCO
PLL Event
Divider
Δ∑-Mod
DSP

Figure 2: The block diagram of the PLL.

```
i0_.CFG_PCLK(net48);
i0_.CFG_SCLK(net023);

//models for test bench configuration
//and monitoring
freq_meter i0_freq_meter("freq_meter");
i0_freq_meter.average_freq(net048);
i0_freq_meter.out(freq_out);
i0_freq_meter.in(VCO_outp);

cfg_data i0_cfg_data("cfg_data");
i0_cfg_data.data(net037);
i0_cfg_data.pclk(net023);
i0_cfg_data.sclk(net48);

clk_gen i0_clk_gen("clk_gen_verilog");
i0_clk_gen.clk_out(net61);

//simulation control
...
```

Listing 4: The overall pll testbench.

3.2 Involving SystemC-AMS in the Concept

While SystemC was initially developed to model digital systems, the SystemC-AMS 1.0 extensions [8] offers the possibility to model analog / Mixed-Signals systems by using of dedicated simulation kernel synchronized with the SystemC event driven kernel [9]. The Timed Data Flow (TDF) [10] class allows the direct implementation of continuous time Laplace transfer functions, which make the implementation of the loop filter of the PLL easier. The Synchronization slows down the simulation about 5 second, as shown in the table 2, therefore a Matlab script has been implemented for translating the analog loop filter transfer function into the equivalent discrete time SystemC model directly.

3.3 Comparison with other HDL

The same system has been modeled in the Verilog-AMS HDL based on the wreal approach shown in [6] in order to evaluate the accuracy of the SystemC model implementation. The overall PLL system runs in the cadence NCSIM digital simulator. A closed loop simulation of the PLL has been run to review the two models regarding the simulation time and phase noise results. The simulation times for 500 μs transient simulation have been compared in the table 2. All PLL models have been implemented according the specifications shown in table 1.

HDLs	Simulation time
SystemC 2.2.0	$31s$
SystemC-AMS 1.0	$36s$
Verilog-AMS(Wreal)	$40s$

Table 2: Simulation execution times

As expected, no significant difference regarding the simulation time can be observed, given the fact that all three simulation are done in the event driven digital simulator completely. Despite its small advantage the SystemC implementation does not require any special simulator nor the AMS-extensions. therefore it can be used in almost any digital centric design flow today.

The phase noise results of the simulation are plotted in the figure 3 and 4, from which it can be observed that the SystemC models exhibits the best simulation efficiency without losing the accuracy of the models considering the PLL phase noise.

Additionally, as previously anticipated in earlier section, the advantages of the SystemC modeling approach lies in the fact that the SystemC modules are based on the C++ classes and offers its object oriented features. A shown in the figure 2, the output signal structure of the PLL can be extended to a base band signal array without any modification of the

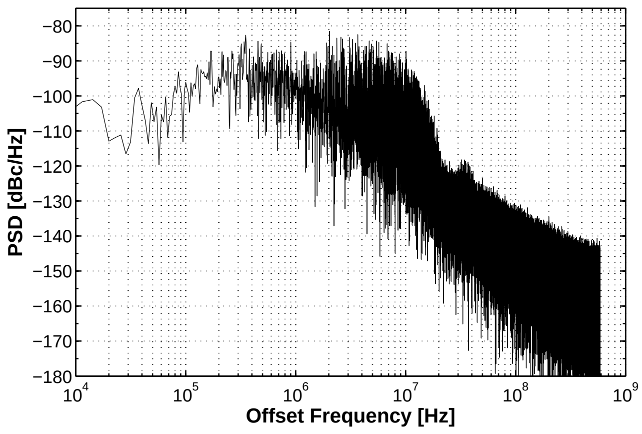

Figure 3: Phase noise simulation result of the SystemC PLL model.

connectivity information, by creating the signal with user defined data types as shown in listing 5.

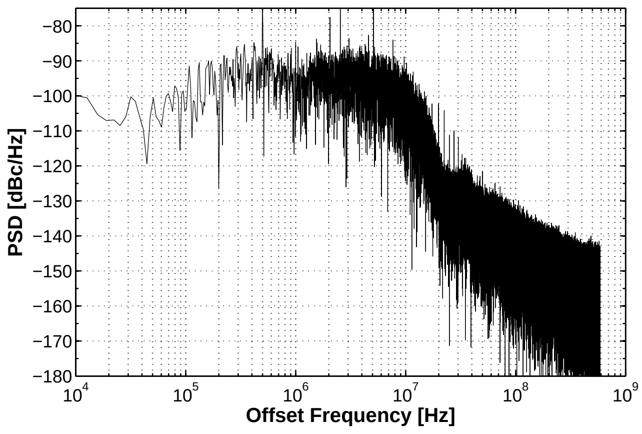

Figure 4: Phase noise simulation result of the Verilog-AMS PLL model.

For the detailed PLL model, the divider is triggered by the pass band VCO oscillation event *VCO_out*, in order to get the accurate response of the equivalent SystemC RF blocks in the PLL. This can be realized by overloading the == operator in the *PLL_SIG* class, which permits only the changes in the *VCO_out* are considered in the sensitive list. Further considerations like the interaction between the PLL output signal and RF base band signal can be resolved by overloading the operators.

```
class BB_SIG{
public:
bool VCO_out
double f0, AI, AQ, PN, deltaF;
BB_SIG(){...}
BB_SIG operator == (const BB_SIG & sig)
const{
return (sig.VCO_out == VCO_out;
}
BB_SIG operator= (const BB_SIG & sig) {
VCO_out=sig.VCO_out;
//f0, AI, AQ, PN, deltaF...
}
//additional operator overloading
...
};
```

Listing 5: Mixed pass band / base band signal type implementation.

For the RF receiver chain, only the part of the PLL output signal that is required in the base band modes are needed, in this case the equivalent base band amplitudes $I(t)$, $Q(t)$, the center frequency f_0 and the instantaneous phase $phi(t)$ describing a potential phase modulation as well as the phase noise present in the LO signal, respectively.

4. CONCLUSIONS AND OUTLOOK

In this paper, we have proposed a method to generate pin accurate SystemC RF models based on the circuit information. This approach has been demonstrated to generate the event driven SystemC models for a fractional-N PLL system. The accuracy and simulation performance of the SystemC models have been evaluated with the results obtained with the event driven Verilog-AMS models. Besides a satisfactory match between the simulation results, the SystemC models exhibits the best simulation performance.

Additionally the multi frequency simulation capability of SystemC can be demonstrated by combining the pin accurate pass band PLL model with other RF base band models without any modification of the connectivity information. The flexibility and portability of an open source C++ based HDL allows us to verify the complete signal path based on the same hierarchical information from schematic level. Thus increases the speed and reduces the error proneness of the model generation process. Furthermore, the proposed method will be applied to a larger and more complex design, which has been discussed in [11], in order to speed up the system design and verification of the digital centric nano-scale CMOS RF system architecture.

5. ACKNOWLEDGMENTS

The author would like to thank Cadence Academic Network (CAN) for the software support used for this work.

6. REFERENCES

[1] T.J. Barnes. Skill: a cad system extension language. In *Design Automation Conference, 1990. Proceedings., 27th ACM/IEEE*, pages 266 –271, 24-28 1990.

[2] SystemC 2.2.0. Open systemc initiative , http://www.systemc.org.

[3] Jesse E. Chen. Modeling RF Systems. *Designers Guide Community*, 2006.

[4] S. Joeres and S. Heinen. Functional verification of radio frequency socs using mixed-mode and mixed-domain simulations. In *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*, 2006.

[5] S. Joeres, H.-W. Groh, and S. Heinen. Event driven analog modeling of RF frontends. In *Behavioral Modeling and Simulation Workshop, 2007. BMAS 2007. IEEE International*, 2007.

[6] Yifan Wang, C. Van-Meersbergen, H.-W. Groh, and S. Heinen. Event driven analog modeling for the verification of pll frequency synthesizers. In *Behavioral Modeling and Simulation Workshop, 2009. BMAS 2009. IEEE*, pages 25 –30, 17-18 2009.

[7] Jesse E. Chen. A modeling methodology for verifying functionality of a wireless chip. In *Behavioral Modeling and Simulation Workshop, 2009. BMAS 2009. IEEE*, pages 96–101, 17-18 2009.

[8] SystemC-AMS 1.0. Standard systemc ams extensions 1.0 , http://www.systemc-ams.org.

[9] Alain Vachoux, Christoph Grimm, and Karsten Einwich. Extending systemc to support mixed discrete-continuous system modeling and simulation. 2008.

[10] OCSI. Standard systemc-ams extension language reference manual. 2010.

[11] N. Zimmermann, B.T. Thiel, R. Negra, and S. Heinen. System architecture of an rf-dac based multistandard transmitter. In *Circuits and Systems, 2009. MWSCAS '09. 52nd IEEE International Midwest Symposium on*, pages 248 –251, 2-5 2009.